CLAIMS

What is claimed is:

1. A method for cross-module in-lining, comprising:
    in a first phase of a compiling process,
        deciding to in-line a first function in a first module into a second function in a second module;
        providing the location of the first function;
        providing instructions for in-lining to be performed in a second phase of the compiling process;
    in the second phase of the compiling process,
        following the instructions to in-line code of the first function into the second function.

2. The method of claim 1 wherein:
    the compiling process comprising a front-end phase, an inter-procedural analysis phase, and a back-end phase;
    the inter-procedural phase being the first phase; and
    the back-end phase being the second phase.

3. The method of claim 1, in the first phase of the compiling process, further having a third function in the module containing the second function.

4. The method of claim 3, in the second phase of the compiling process, further getting rid of the third function in the module containing the second function after using that third function to in-line its code into the second function.

5. The method of claim 4 wherein the third function being selected from a group consisting of the first function and a clone of the first function.

6. The method of claim 1, wherein, in the second phase of the compiling process, in-lining the code of the first function into the second function uses a clone of the first function.

7. The method of claim 1, wherein, in the second phase of the compiling process, the code used to be in-lined into the second function is stored in a file.

8. The method of claim 1 wherein, in the second phase of the compiling process, the code used to be in-lined into the second function is stored in a library.

9. The method of claim 1 wherein the instructions include at least a list of callees to be in-lined and corresponding callers.

10. A method for compiling a first set of modules having programming source code, comprising:

in a first phase,

from the first set of modules, providing a second set of modules having first intermediate representations;

in a second phase,

performing in-line analysis on the second set of modules;

providing instructions for in-lining to be performed in a third phase of the compiling process; and

providing a third set of modules having second intermediate representations optimized from the first intermediate representations;

in the third phase of the compiling process,

following the instructions to perform in-lining, and

providing a fourth set of modules having third intermediate representations optimized from the second intermediate representations.

11. The method of claim 10, in the second phase, further using code in the module containing a function caller of a function callee to transform in-lining.

12. The method of claim 11 wherein the code being selected from a body of the function callee.

13. The method of claim 11 wherein the code being selected from a clone of the function callee.

14. The method of claim 10 wherein the instructions include at least one of:

a set of function caller including at least one function caller;

a set of function callee including at least one function callee;

the order for transformation of in-lining;

the location of at least one function callee; and

decisions whether to keep a body of at least one function callee after in-lining transformation.

15. A computer-readable medium embodying a compiler, the compiler comprising:

a front-end phase;

a cross-module analysis phase; and

a back-end phase;

wherein

the front-end phase invokes the cross-module analysis phase;

the cross-module analysis phase

determines whether a callee is to be in-lined into a caller in the back-end phase;

provides instructions for the back-end phase to transform in-lining code of the callee; and

invokes the back-end phase; and

the back-end phase

transforms the in-lining code based on the instructions.

16. The computer-readable medium of claim 15 wherein the back-end phase further performs tasks related to in-lining.

17. The computer-readable medium of claim 16 wherein the tasks related to in-lining include at least deleting the callee in a module containing the caller.

18. The computer readable medium of claim 15 wherein transforming the in-lining code uses code of a clone of the callee.

19. The computer-readable medium of claim 15 wherein a call to the callee is in a module that does not include the callee.

20. The computer-readable medium of claim 15 wherein the instructions include at least a list of callees.